Mississippi State UNIVERSITY

Center for Air Sea Technology

DESIGN DOCUMENT

for the

SURFACE CURRENTS DATA BASE (SCDB) MANAGEMENT SYSTEM (SCDBMS) Version 1.0

Technical Note 08-94

AUGUST 1994

Prepared for: Naval Oceanographic Office, Stennis Space Center, Mississippi
Contract Number: NAS 13-330 / Order No. 53

Mississippi State University Center for Air Sea Technology
Building 1103, Stennis Space Center, MS 39529-6000

DESIGN DOCUMENT

and DATABASE SPECIFICATION

FOR THE

SURFACE CURRENTS DATABASE MANAGEMENT SYSTEM (SCDBMS)

SCDBMS Version 1.0 (29 August 1994)

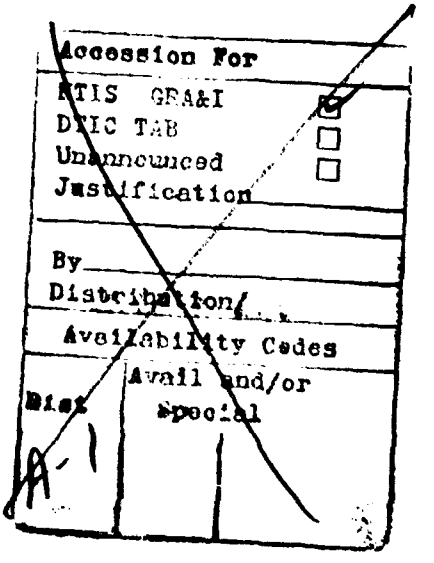
CONTRACT NO: NAS 13-330, Order No. 53

Prepared for:

NAVAL OCEANOGRAPHIC OFFICE
STENNIS SPACE CENTER, MS 39529

Prepared by:

Mississippi State University
Center for Air Sea Technology
Building 1103, Room 233
Stennis Space Center, MS 39529-6000



Acknowledgements

UNIX is a trademark of American Telephone and Telegraph (AT&T), Incorporated

SUN, SUNOS and SparcStation are trademarks of Sun Microsystems, Incorporated

Motif is a trademark of the Open Software Foundation

X-Windows is a trademark of the Massachusetts Institute of Technology

Empress is a trademark of Empress Software, Incorporated

UNIRAS ag/X Toolmaster is a trademark of Advanced Visual Systems, Incorporated

Contributors to the SCDMS Software Design Document are:

Ms. Cheryl Cesario, Project Manager
Mr. Ramesh Krishnamagaru, Senior Software Engineer
Mr. Vishnu Mohan Das, Programmer
Mr. Micheal S. Foster, Graphics and Editing

Table of Contents

Page Number

LIST OF FIGURES:

SCDBMS SOFTWARE DESIGN DOCUMENT

1. SCOPE

1.1. *Identification*

Computer Software Configuration Item (CSCI): Surface Currents Database Management System (SCDBMS)

Version: 1.0

Release Date: 23 August 1994

Contract No: NAS 13-330, Order No. 53

Contractor: Mississippi State University
Center for Air Sea Technology
J. H. Corbin, Director
Building 1103, Room 233
Stennis Space Center, MS 39529-6000
Telephone: (601) 688-2561
Facsimile: (601) 688-7100

Project Manager: Ms. Cheryl Cesario
Mississippi State University
Center for Air Sea Technology, Stennis Space Center, Mississippi 39529-6000. (MSU Proposal No. 93-3-467)
Telephone: (601) 688-7141
Facsimile: (601) 688-7100

1.2 Overview

The Surface Currents Database Management System (SCDBMS) provides access to the Surface Currents Data Base (SCDB) which is maintained by the Naval Oceanographic Office (NAVOCEANO). The SCDBMS incorporates database technology in providing seamless access to surface current data. The SCDBMS is an interactive software application with a graphical user interface (GUI) that supports user control of SCDBMS functional capabilities.

1.3 Document Overview

The purpose of this document is to define and describe the structural framework and logical design of the software components/units which are integrated into the major computer software configuration item (CSCI) identified as the SCDBMS, Version 1.0. The preliminary design is based on functional specifications and requirements identified in the governing Statement of Work prepared by the Naval Oceanographic Office (NAVOCEANO) and distributed as a request for proposal by the National Aeronautics and Space Administration (NASA). The content and format of this document are specified by Department of Defense (DOD)-STD 2167A (2.1).

Appendix A contains a glossary of terms used within this document. Appendix B is a listing of acronyms used within this document. The SCDBMS relational database specification is contained in Appendix C. Appendix D contains functional and design requirements for the

SCDBMS relational database management system (RDBMS). Refer to Appendix E for a description of the SCDBMS development environment. Appendix F contains listings of relevant source code structures. Appendix G provides information about the SCDBMS configuration file.

2 REFERENCED DOCUMENTS

Note: The section(s) or subsection(s) of this document wherein a reference is cited appears as parenthetical information following each document listed in this section.

2.1. DOD-STD 2167A "Defense System Software Development", AMSC No. N4327, 29 Feb 88. (1.3)

2.2. Young, Douglas A., "The X Windows System Programming and Applications with Xt, OSF/Motif Edition", Prentice Hall, Englewood Cliffs, NJ, 1990. (3.1.1.1)

2.3. Jurkevics, Andrew, "Database Design Document for the Naval Environmental Operational Nowcast System, Version 3.5", Naval Oceanographic and Atmospheric Research Laboratory, Monterey, CA, 1 June 1992. (3.1)(3.1.1.4)

2.4. Documentation (Series) for the Empress Relational Database Management System, Version 6.X, Vol. A1-D1, Empress Software Incorporated, Greenbelt, MD, 1993. (3.1.1.4)

2.5. Users Manual and Reference Guides for UNIRAS ag/X Toolmaster Graphics Extensions Library, Version 6v3b, UNIRAS, Incorporated, Overland Park, KS, 1993. (3.1.1.1)

2.6. Coffin, Stephen, "UNIX System V Release 4: the Complete Reference", Osborne McGraw-Hill, New York, NY, 1990. (Useful reference for broad overview)

3 PRELIMINARY DESIGN

3.1 CSCI *Overview*

The Surface Currents Database (SCDB) Management System (SCDBMS), a stand-alone system, is the major Computer Software Configuration Item (CSCI) to be developed by this project. Functional requirements, as identified by the sponsor, were not defined in the context of a modular approach to software development. Therefore, the tasks necessary to fulfill the functional requirements have been divided among/assigned to the internal Computer Software Components (CSC) for accomplishment. The top-level (simplistic) SCDBMS architecture is illustrated in Figure 1.

There are three principle CSC's within the SCDBMS structure:

a. **CSC-1: Graphical User Interface (GUI)** - incorporates initialization, window management, user interface and display functionality;
b. **CSC-2: Data Management Module (DMM)** - provides functional data management using relational RDBMS technology;
c. **CSC-3: Data Administration Module (DAM)** - incorporates administrative control of the application and is inaccessible to the general user.

The only access to the SCDBMS is via the **User-GUI** external interface which supports 1) application initialization, 2) user control of the SCDBMS using interactive techniques, 3) response/feedback to the user in the form of data display (graphical or numerical) and status indicators, 4) indirect access to data contained in the SCDB, and 5) indirect access to files and data that are not resident within the SCDB. The DMM accesses the SCDB and data/configuration files through its external interfaces. The Naval Environmental Operational Nowcast System (NEONS)(2.3) controls access to the external SCDB and has been modified to accommodate specific storage and retrieval requirements of the SCDBMS.

3.1.1 CSCI Architecture

Figure 1 illustrates the SCDBMS top-level (simplistic) module and its external interface architecture.

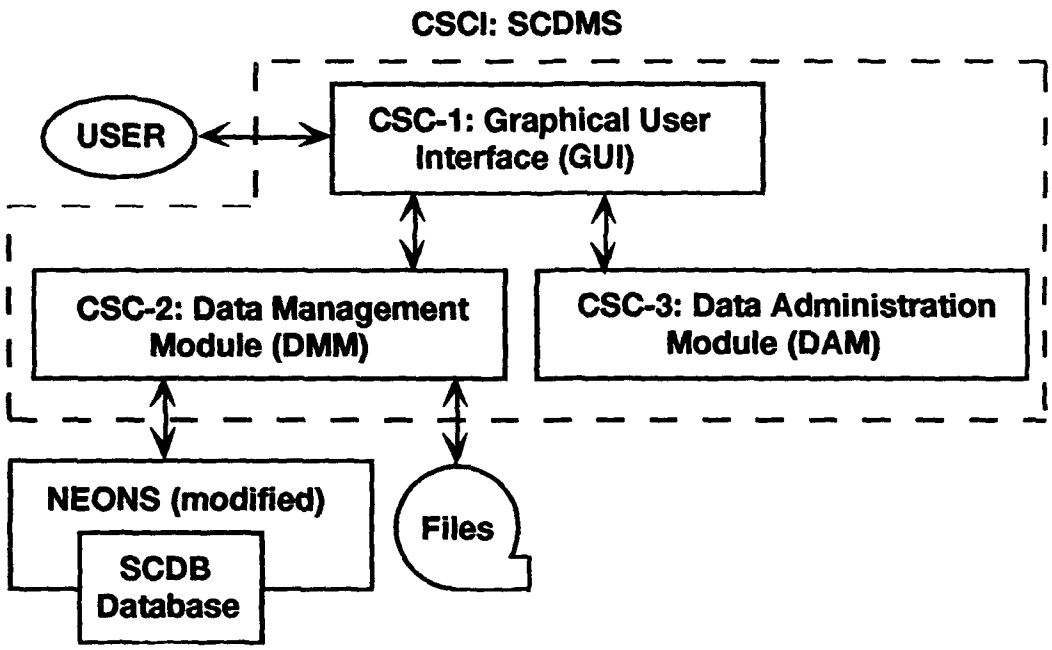


Figure 1. Top-level modular structure of the Surface Currents Database Management System (SCDBMS). The SCDBMS is the Computer Software Configuration Item (CSCI). There are three Computer Software Components (CSC).

3.1.1.1 CSC-1: Graphical User Interface (GUI). The SCDBMS GUI supports and manages the link between the user and the SCDBMS. Through the GUI, the user exercises all available SCDBMS control options. The SCDBMS provides display windows and dialogs to the user for interpretation and/or interactive response. The GUI is integrated with the following non-developmental software:

a. **X-Windows** (proprietary): Manages all controls and display windows. The X-Windows client/server model supports remote user access to the SCDBMS using network protocol via the UNIX operating system. X-Windows operation is explained in detail by reference 2.2.

b. **Open Software Foundation (OSF) Motif Widget Toolkit** (proprietary): Provides a widely accepted standard inventory of window components (widgets) that is pleasing to the eye and easy to interpret. For more information on programming using the Motif Widget Toolkit, see reference 2.2.

c. **UNIRAS ag/X Toolmaster** (proprietary): Handles graphical representation and visualization of data displays. See reference 2.5 for detailed information concerning UNIRAS ag/X Toolmaster.

Functionally, the GUI:

a. Performs application initialization and event monitoring functions;
b. Exercises direct, centralized control over the DMM and DAM;
c. Monitors activities of the DAM and the DMM and its database/file interfaces;
d. Intercepts, interprets and routes user interactive commands;
e. Serves as the agent for communications between the DMM and the DAM;
f. Provides status information and feedback to the user;
g. Provides the windowing environment for visualization of data, display of control elements and intercepting user interactive commands;
h. Receives and interprets user input via the keyboard or mouse pointing device;
i. Incorporates visualization routines to plot data displays for viewing by the user.

3.1.1.2 CSC-2: Data Management Module (DMM). The SCDBMS DMM is primarily tasked with managing access to and communications with the external SCDB via its NEONS (modified) interface. The DMM communicates with the GUI to pass information from the database and external files. It communicates with the DAM to enable SCDB Administrator (SCDBA) control over application configuration. When a data request is received from the GUI, the DMM notifies NEONS to retrieve the data from the SCDB. When the data has been received, the DMM informs the GUI and passes its location in memory. The DMM also receives data management instructions from both the GUI (user) and,indirectly, the DAM (SCDBA). In turn, through NEONS,the DMM translates these instructions into Structured Query Language (SQL) commands to the RDBMS. The DMM is also responsible for allocating the internal memory space for data retrieved from the database or data destined for database ingestion.

3.1.1.3 CSC-3: Data Administration Module (DAM). The SCDBMS DAM is responsible for managing SCDBA functions within the SCDBMS. SCDBA functions are not accessible to the general user. These functions include importing data into the database and updating of SCDBMS application tables (classification codes and data source codes) which are then made available for user interaction via the GUI.

3.1.1.4 NEONS (modified). The SCDB is accessed via a modified version of NEONS, which performs high level management of the SCDB. The specific modifications to NEONS for the SCDBMS allow use of additional keys to support optimal database query and data retrieval from the SCDB primary database tables (which contain the actual data). The SCDB is a relational database. NEONS includes both the structural model for the database and the applications programmer interface (library routines) required to access the SCDB tables using the American National Standards Institute (ANSI) standard SQL. NEONS was developed by the Naval Research Laboratory, Monterey, CA. A detailed description of NEONS is available in the NEONS design document (2.3). NEONS employs the proprietary Empress (2.4) relational data management system (RDBMS) for low-level management of the SCDB.

3.1.1.5 USER-GUI (USER/CSC-1) External Interface. The USER-GUI interface provides user control of the SCDBMS. The interface is implemented by means of (1) a pointing device (mouse) which is employed by the user to pass signal instructions (events) to the application, and (2) the keyboard which is used to pass text information to the application.

3.1.1.6 GUI-DMM (CSC-1/CSC-2) Internal Interface. The GUI-DMM internal interface consists of all facilities for communication between the GUI and the DMM. The GUI-DMM interface is an abstraction employed to group those actions which pass information between the two modules. Actual communication between the GUI and the DMM is usually accomplished directly by the GUI and DMM functions involved.

3.1.1.7 GUI-DAM (CSC-1/CSC-3) Internal Interface. The GUI-DAM internal interface consists of all facilities for communication between the GUI and the DAM. The GUI-DAM interface embodies the same functionality and abstract character as the GUI-DMM interface.

3.1.1.8 DMM-DAM (CSC-2/CSC-3) Internal Interface. There are only two instances where the DMM and DAM communicate directly with each other. These instances are discussed in Section 4.2.2. Otherwise, the DMM and DAM communicate indirectly via the GUI.

3.1.1.9 DMM-NEONS External Interface. The DMM-NEONS external interface consists of all NEONS library functions that are integrated within the SCDBMS. These functions generate SQL commands to the Empress RDBMS, returning status indicators and performing two-way data transferal, when successfully executed.

3.1.1.10 DMM-FILES External Interface. The DMM-FILES external interface consists of all functions within the DMM which result in direct reading from or writing to files within the operating system's file management facility.

3.1.2 System States and Modes

The SCDBMS is an interactive software application. The application is always in an event-driven state. As with all event-driven software applications, the SCDBMS may assume either of two execution states (modes): (1) processing mode and (2) rest (idle) mode. The SCDBMS default state is the rest mode. When not processing data in response to user input, the application automatically reverts to the rest mode (event monitoring loop) and awaits the next user command or input.

3.1.3 Memory and Processing Time Allocation

The interactive, event-driven nature of the SCDBMS precludes a quantitative description of memory allocation and processing time among SCDBMS CSC's. It is therefore considered sufficient to state that the SCDBMS responds to user-generated commands by allocating memory, swap space and processor time within the limitations imposed by available memory, process loading and UNIX operating system constraints.

3.2 CSCI Design Description

The CSCI (SCDBMS) consists of three CSC modules with functionality as described above. Functional requirements for the CSCI, as stated by the sponsor, could not always be accomplished wholly and completely within a single CSC. For instance, the final display of profile

data requires the cooperative contribution of both the GUI and the DMM modules. The remainder of this section identifies SCDBMS requirements, by CSC, and describes the software design employed to achieve the required functionality. The CSCI incorporates all SCDBMS functional requirements (SFR) and design requirements (SDR) as presented in Section 7 of this document. In addition, the CSCI achieves specific design requirements which are not accomplished by any CSC (SDR1 through SDR4).

3.2.1 Graphical User Interface (GUI) (CSC-1)

CSC-1 is responsible for providing the visual display link between the user and the SCDBMS main interactive display. The design for the SCDBMS main interactive display window is illustrated in Figure 2. The GUI is the ultimate destination of all visible computational results achieved by the CSCI and its subordinate CSC's. The DMM (CSC-2) performs data management chores for the GUI and DAM. The DAM (CSC-3) performs SCDBA applications control and maintenance functions. The GUI and its subordinate components achieve the following specific functional and design requirements, either wholly or in concert with other CSC's (see Section 7):

SCDBMS Functional Requirements (SFR)

SFR1, SFR2, SFR3, SFR4, SFR5, SFR6, SFR7, SFR8, SFR9, SFR10
SFR11, SFR12, SFR13, SFR14, SFR15, SFR17, SFR18, SFR19, SFR 20,
SFR21, SFR22, SFR23, SFR24, SFR26, SFR30, SFR32, SFR33.

SCDBMS Design Requirements (SDR)

SDR8

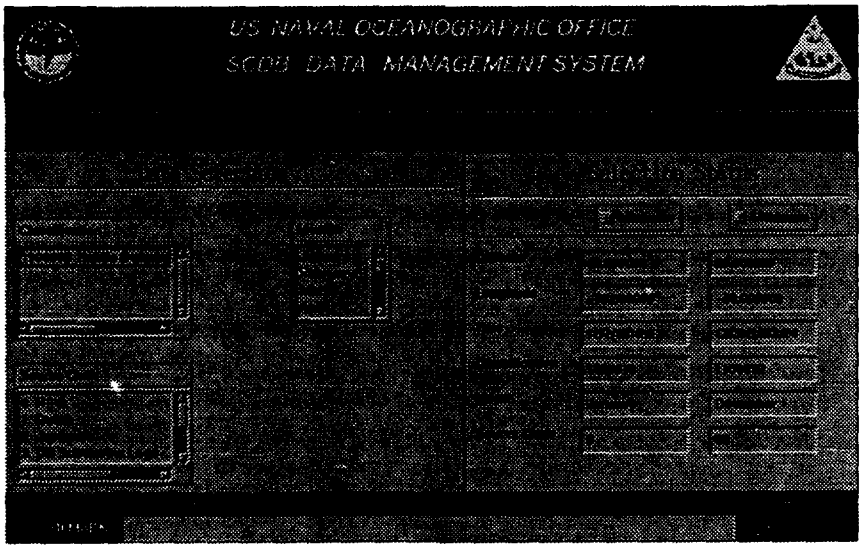


Figure 2. Illustration of the SCDBMS Graphical User Interface (GUI) (CSC-1) display screen. The GUI is the origin and final destination for all interactive coordination between the user and user-controllable processes resident in the Data Management Module (DMM) (CSC-2), the Data Administration Module (DAM) (CSC-3) and their internal/external interfaces.

3.2.2 Data Management Module (DMM) (CSC-2)

The Data Management Module (DMM) (CSC-2) controls the acquisition and distribution of data within the SCDBMS CSCI. The user has no direct interaction with the DMM. The DMM and its sub-level components achieve the following specific functional and design requirements, either wholly or in concert with other CSC's (see Section 7):

SCDBMS Functional Requirements (SFR)

SFR13, SFR14, SFR15, SFR17, SFR18, SFR19, SFR20, SFR21, SFR22, SFR23, SFR24, SFR27, SFR28, SFR29, SFR30, and SFR31.

SCDBMS Design Requirements (SDR)

SDR5, SDR6, and SDR7.

3.2.3 Data Administration Module (DAM) (CSC-3)

The DAM provides tools that enable application maintenance by the SCDBA. There are no specific design requirements for the DAM. The DAM and its sub-level components achieve the following specific functional requirements, either wholly or in concert with other CSC's (see Section 7):

SCDBMS Functional Requirements (SFR)

SFR12, SFR13, and SFR15.

4 DETAILED DESIGN

Figure 3 illustrates the functional connectivity within the SCDBMS and forms the basis for the detailed design of the SCDBMS CSCI. The preliminary design (see Section 3) provides a broad overview of the framework and functional design of the SCDBMS CSCI and its three major CSC's [Graphical User Interface (GUI), Data Management Module (DMM) and Data Administration Module (DIM)]. In this section, each CSC and its computer software units (CSU) will be described in greater detail.

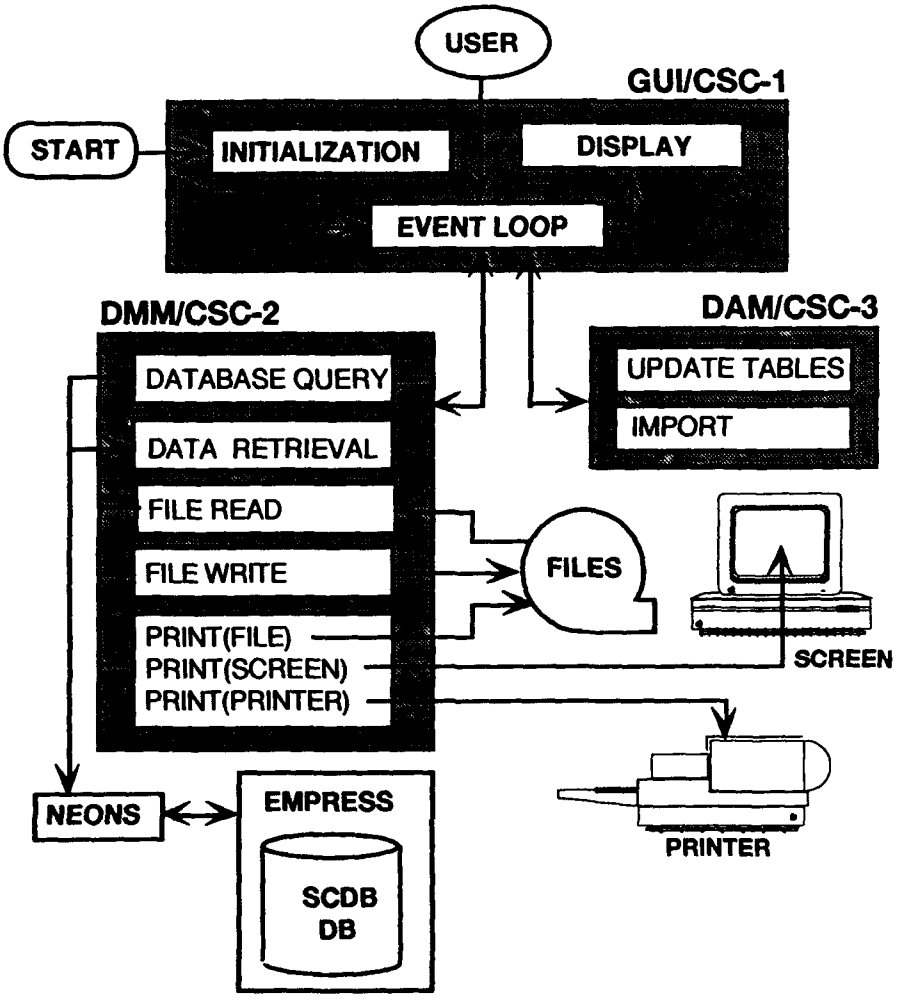


Figure 3. SCDBMS detail design and connectivity.

4.1 CSC-1 - The SCDBMS Graphical User Interface (GUI)

The SCDBMS GUI (CSC-1) embodies all facilities for user interaction. Functions performed by the GUI include:

a. Application initialization;
b. Event monitoring;
c. Window management;
d. Communicating with the user and the Data Management and Data Administration modules;
e. Creation and display of data plots from data received from the DAM;
f. Processing and interpretation of user commands via the pointing device (mouse) and the keyboard;
g. Providing visual feedback in response to user input and internal messages;
g. Management of the user "Help" facility;
g. Application shutdown.

4.1.1 Initialization

The Initialization CSU receives control when the operating system releases control to the application. Functions performed by the Initialization CSU include:

a. Parsing and interpreting any command line arguments that identify the default configuration filename by the function **parseArg()**;
b. Establishing a connection to the X server, initializing the Intrinsics layer and obtaining a TopLevelShell widget by the function **XtInitialize()**;
c. Obtaining pointers to the Xlib Display and Screen structures using the functions **XtDisplay()** and **XtScreen()**;
d. Reading the SCDB configuration file;
e. Opening the database using the **db_start()** function;
f. Creation and display of the SCDBMS Main Window, its window tree and data structures including default range values obtained from the configuration file via the function **CreateTree()**;
g. Initializing internal SCDB data structures via function **InitMoodsStruct()**;
h. Establishing user access privileges.
i. Executing the **XtMainLoop()** event monitoring function which dispatches all subsequent events to registered event handlers and/or callback functions.

4.1.1.1 SCDBMS Initialization Specification/Constraints

The design criteria for the SCDBMS Initialization CSU are listed in Section 7, "Requirements Traceability". This CSU is constrained by the following software-specific requirements:

a. X-Windows and OSF Motif must be installed and accessible;
b. The Empress RDBMS must be installed and accessible;
c. UNIRAS ag/X Toolmaster must be installed and accessible;
d. NEONS must be installed and accessible;
e. A SCDB must be available.

4.1.1.2 SCDBMS Initialization Design

The SCDBMS Initialization design adopts the standard X-Windows/OSF Motif protocol. The **XtMainLoop()** function ultimately assumes control after all initialization procedures have been accomplished, including display of the Main Window. **XtMainLoop()** implements an infinite event-monitoring loop which executes (dispatches messages to) registered event handling and/or callback functions in response to the occurrence of events.

4.1.2 SCDBMS Main Window

The Main Window CSU is illustrated in Figure 2. It is subdivided into five areas:

a. The title area;
b. The menubar which allows access to user-selectable menu items via pull-down menus;
c. The "Data Selection" area which provides access to most configuration parameters required to identify SCDB datasets and/or subsets;

d. The "Selection Status" area which displays range limits (minimum and maximum) for parameters defined within the "Data Selection" area and the "Select: Region" and "Select: Time" menu items;
e. The "Remark" textbox which communicates important messages, such as errors, to the user.

The Main Window is created and displayed during application initialization. All SCDBMS features are initiated via the Main Window or its subordinate windows (dialogs, textboxes, listboxes, etc.) which appear in response to user interaction with Main Window selection features.

4.1.2.1 SCDBMS Main Window Specification/Constraints

The design criteria for the SCDBMS Main Window CSU are listed in Section 7, "Requirements Traceability". This CSU is constrained by the following requirements:

a. The X-server must be opened and available for display;
b. UNIRAS ag/X Toolmaster must be active for handling graphics functionality;
c. The Empress RDBMS must be active and the SCDB must be opened and available for reading.

4.1.2.2 SCDBMS Main Window Design

The SCDBMS Main Window design employs the X, Motif, NEONS, and UNIRAS ag/X Toolmaster libraries. The design is implemented during initialization by the function **CreateTree()**.

4.1.3 GUI Functionality

After the SCDBMS Main Window has been displayed, the user is free to interact with it using the mouse and keyboard. The GUI translates all mouse and keyboard events into calls to event handlers and/or callback functions. If the event requires retrieval/ingestion of data from/to the SCDB, the GUI calls upon a function within the DMM. If the event requires SCDBA intervention or a security check, the GUI calls upon a function within the DAM.

4.1.3.1 The Menubar

The menubar consists of five pull-down menu headers (**Select, MDBA, Products, Util**), each of which offers two or more subordinate menu items, plus the **Help** button which provides access to the on-line help facility. When clicked, a menu header is programmed to display its subordinate menu items, which themselves may be selected by clicking. The reaction of the application to selection of a menu item depends upon the functionality of the particular menu item. Selections made via the menubar provide access to all options that are not directly associated with establishing the criteria for data retrieval.

4.1.3.1.1 Select: Region

When the "Region" menu item is clicked, the function **regionButtonPress()** is executed and the window illustrated in Figure 4 is displayed. This window offers three options for selecting the boundaries of a geographical region. The "Region List" contains a listing of user-selectable pre-defined regions. When a pre-defined region is selected from the "Region List", the region boundary is displayed within the map window as a dashed rectangle, and, the labeled textboxes within the "Region Coordinates" widget are modified to hold the latitude and longitude coordinate

limits of the region. When a region is selected by dragging the mouse cursor across an area of the map display, the "Region Coordinates" textboxes are modified to hold the latitude and longitude coordinates of the region. Finally, a region may be defined by selecting individual textboxes within the "Region Coordinates" widget and entering replacement values from the keyboard. If greater resolution is required, the "Zoom" button located below the map will produce a pop-up window containing an enlarged image of the region defined on the map. The "File" pull-down menu provides menu items to exit the display and to reset the coordinates to their original values. Selecting the "Exit" menu item destroys the window and sends the latitude and longitude boundary coordinates to the "Selection Status" board within the SCDBMS Main Window.

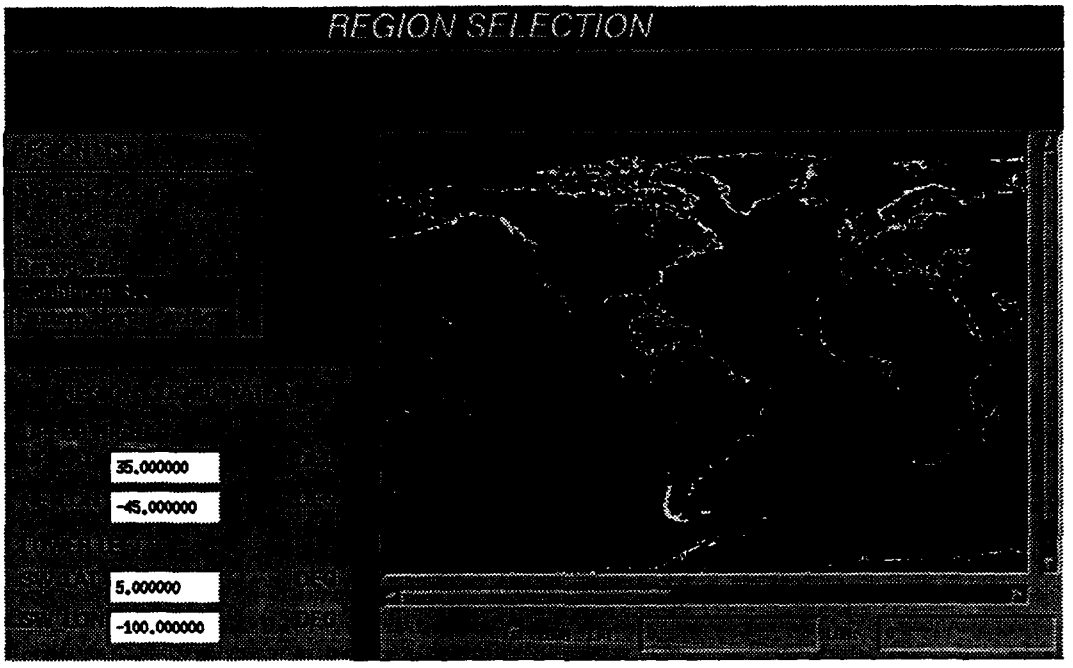


Figure 4. The SCDBMS Region Selection Window. This window appears in response to selecting the "Region" menu item via the "Select" pull-down menu.

4.1.3.1.2 Select: Time

When the "Time" menu item is clicked, the function **timeButtonPress()** is executed and the "Time Selection" window illustrated in Figure 5 is displayed. This window offers options for selecting the time constraints for a SCDB query. If the "Start" button is activated, the adjacent row of text widgets is also activated for input of minimum date/time values. If the "End" button is activated, the adjacent text widgets are activated for entry of maximum date/time values. Date entries may also be changed using the up or down arrow switch buttons to the right of the vertically arrayed "YEAR", "MONTH", "DAY", and "JULIAN" textboxes. These arrow switches are manipulated by clicking. Changes made using the arrow switches are applied to the activated "Start" or "End" row of textboxes. Clicking the "Exit" button closes the "Time Selection" window and transfers the values to the maximum and minimum Date/Time textboxes within the "Selection Status" board of the SCDBMS Main Window.

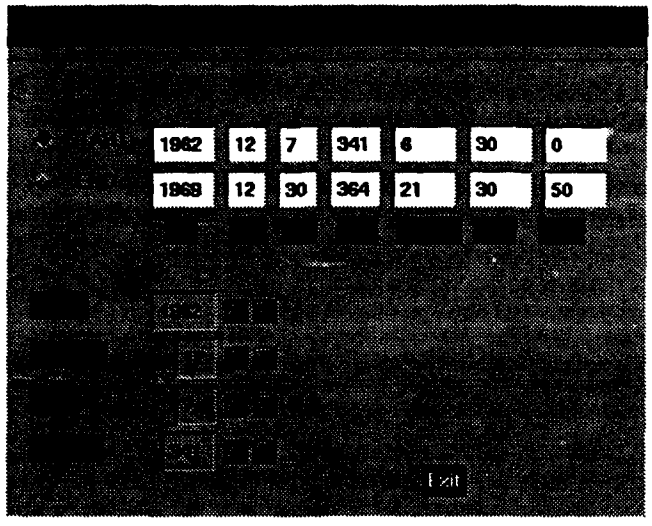


Figure 5. The SCDBMS Time Selection Window. This window appears in response to selecting the "Time" menu item via the "Select" pull-down menu.

4.1.3.1.3 Select: Data

When the "Data" menu item is selected, the function **dataSourceDialog()** is executed, which produces a "source dialog" window with option buttons for selecting the direct source of data. Pressing the "Data" button activates the default function **dbaseTogglePress()**, which establishes the SCDB as the source for surface current data. Pressing the "File" button activates the function **fileTogglePress()** which, in turn, activates the function **selectDataFile()**. **selectDataFile()** displays the "Load Defaults File" dialog window. This dialog window contains the necessary listboxes, textboxes and buttons to support user selection of an input data file. When finished with path/filename entry and file loading, clicking the "Exit" button executes the function **destroyDialog()** which destroys the "Load Defaults File" dialog window. When the "source dialog" window's "Ok" button is pressed, the function **destroyDialog()** is again called to destroy the "source dialog" window and return the SCDBMS to the **XtMainLoop()** event monitor.

4.1.3.1.4 Select: Exit

When the "Exit" menu item is selected, the function **exitProgram()** is executed. Clicking the "Exit" menu-item closes the SCDBMS main display screen and returns the user to the system command line prompt.

4.1.3.1.4 MDBA:Import (SCDBA Only)

During initialization, the function **createDbaTools()** determines if the user is the SCDBA. The source code. . .

if (!(isDba)) { XtSetArg (wargs[n], XmNsensitive, False); }

sets the "Import" menu item's sensitivity argument to "False", thereby disallowing data import by users other than the SCDBA. When the SCDBA selects the "Import" menu item from the "MDBA" pull-down menu, the function **createImportExportDialog()** is executed with the

importexport->type variable set to **IMPORT**. The "Input File" pop-up window (Figure 6) is then displayed, providing a textbox for entry of the file name (including path, if needed) and "Ok" and "Exit" buttons. To be ingested, the file contents must be in the "SCDB Admin" format. Clicking the "Ok" button executes the function **importexportOkSelect()** which, in turn, calls the DMM function **readIngest_file()** to read the file and ingest the data into the database. Clicking the "Exit" button executes the function **importexportExitSelect()** which destroys the "Import File" pop-up window and returns to the **XtMainLoop()** function without performing any further data ingestion operations.

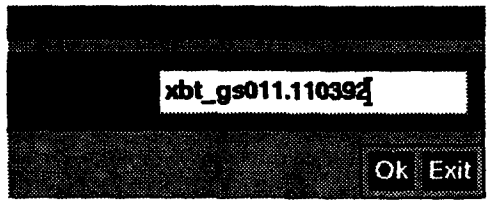


Figure 6. MDBA Filename Entry Dialog Window for Importing Files.

4.1.3.1.5 MDBA:Export

The SCDB export facility is available to all users possessing a level of access appropriate for the data being exported. When the user selects the "Export" menu item from the "MDBA" pull-down menu, the function **createImportExportDialog()** is executed with the **importexport->type** variable set to **EXPORT** and the "Export File" pop-up window (Figure 7) is displayed. The export pop-up window contains a textbox for entry of the export file name (including path, if needed), and "Ok" and "Exit" buttons. Clicking the "Ok" button executes the function **importexportOkSelect()** which, in turn, calls the DMM function **exportData2File()** which writes the data to the named file in "SCDB Admin" format. Clicking the "Exit" button executes the function **importexportExitSelect()** which destroys the file export pop-up window and returns to the **XtMainLoop()** function without performing any further data export functions.

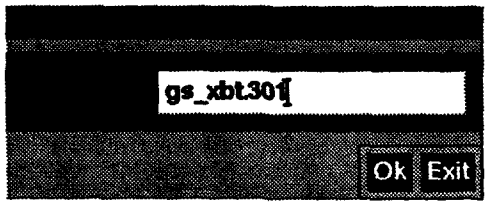


Figure 7. MDBA Filename Entry Dialog Window for SCDB "Admin" Export.

4.1.3.1.6 MDBA:Update Table (SCDBA Only)

The "Update Table" menu item is a tool for use of the SCDBA only. During initialization, the function **createDbaTools()** determines if the user is the SCDBA. The source code...

if (!(isDba)) { XtSetArg (wargs[n], XmNsensitive, False);}

sets the "Update Table" menu item's sensitivity argument to "False", thereby disallowing data table updates by users other than the SCDBA. When selected, the "Update Table" menu item produces a

submenu with two selectable menu items: "Update Source"; and "Update Classification". The GUI functions respectively responsible for these options are **sourceUpdate()** and **updateClassification()**, producing the pop-up windows shown in Figures 8.a and 8.b. Each of these functions executes the DAM function **changeTable()** to "ADD", "DELETE", or "UPDATE" the contents of the specified qualitative code. **changeTable()**, in turn, calls the DMM function **moodsUpdateQltvInfo()** which updates the SCDB qualitative information table and makes an appropriate entry in the SCDB Log.

Figure 8.a. MDBA Update Source Pop-up Window.

Figure 8.b. MDBA Update Classification Pop-up Window.

4.1.3.1.7 PRODUCTS: Inventory

The "Inventory" menu item performs no action in the current implementation of the SCDBMS. It is a "hook for future modification to the SCDBMS. When the user selects the "Inventory" menu item, there is no response. No functions are registered with this button.

4.1.3.1.8 Products: Plots

When the "Plots" menu item is selected from the "Products" pull-down menu, the GUI function **plotDialog()** is executed. **plotDialog()** directly executes the function **createProductDialog()**, which creates and displays the "label" pop-up dialog window shown in Figure 9. Numerous functions and callbacks are required in support of the "label" dialog. This section describes those that perform key roles in generating SCDBMS plot and report products.

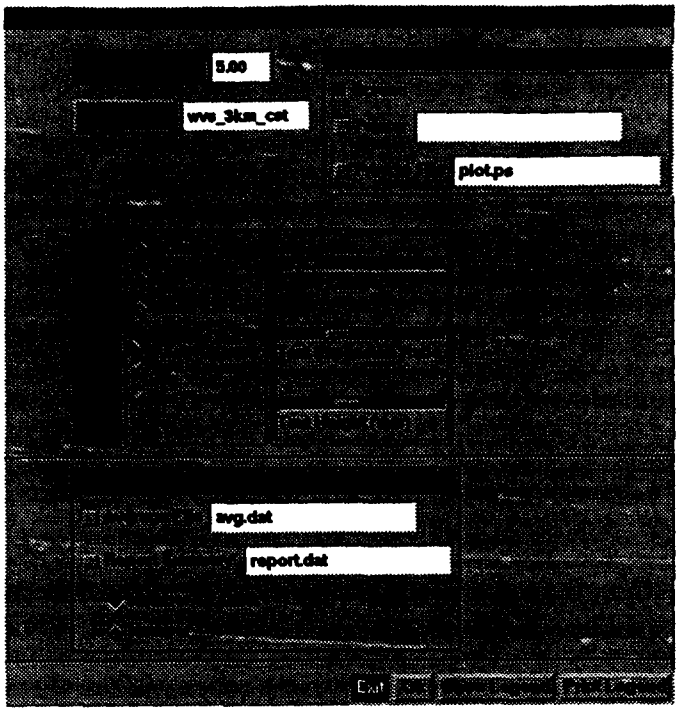


Figure 9. The Products "Plots" Window.

Resolution Selection: The callback function **getLonDeg()** obtains the resolution of the plotting subareas from the textbox labelled "Resolution(deg)". This information is stored in the **scdb_struct** elements **scdb_struct->lon_text**, **scdb_struct->data->xdelta** and **scdb_struct->data->ydelta** where it can be accessed by plot and report generation functions. This function determines the size (in degrees) of the areas in which data will be grouped for processing and display. **scdb_struct** is a C structure of type **NAVO** (see Appendix F).

Coastline Selection: The function **createCoastlineMenubarOptions()** produces a pull-down menu containing four buttons for coastline selection, "3km, "8km", "20km" and "NoCoast". These buttons are linked to registered callback functions **get3km()**, **get8km()**, **get20km()** and **getNoCoast()**. Each of these callbacks copies the appropriate text string into **scdb_struct->coast_type**.

Print Options: The "Print Options" widget is created by **createProductDialog()** and consists of three buttons, "Screen", "Printer" and "Print to File", plus textboxes for entry/display of the printer definition and the name of the file to be written for later printing.

Screen - When the "Screen" button is selected, the callback function **getScreen()** is executed, which assigns output to the widget identified in **scdb_struct->data->screen**.

Printer - When the "Printer" button is selected, the callback function **getPrinterName()** is executed, which obtains the printer name stored in the widget **scdb_struct->data->printer**. A default printer name is assigned during dialog creation but may be changed by activating the adjacent textbox for keyboard entry. The callback function **getPrinter()** acquires the printer name and stores it in the widget as a text string.

Print to File - When the "Print to File" button is selected, the callback function **getPrintFileName()** is executed, which obtains the name of the target file to create as the target for product output. A default file name is assigned to the widget **scdb_struct->data->printFile**

during dialog creation but may be changed by activating the adjacent textbox for keyboard entry. The callback function **getPrintFile()** obtains any new file name entered in the textbox and stores it in the widget as a text string.

Plot Selection: There are six plot product options available. Each option is tied to its own button within the "Plots" section of the "label" pop-up dialog window. Only one plot option button may be selected at a time. When selected, each button executes a callback function which assigns an integer value to **scdb_struct->data->plot_type**. The functions and the corresponding integer values assigned to **scdb_struct->data->plot_type** are as follows:

getStability() assigns 1 to reflect selection of the "Stability" button,
getMaxCurrent() assigns 2 to reflect selection of the "Max. Current" button,
getMeanVector() assigns 3 to reflect selection of the "Mean Current" button,
getNoOfObs() assigns 4 to reflect selection of the "No. of Obs." button,
getAveScalar() assigns 5 to reflect selection of the Ave. Scalar" button, and
getNoPlot() assigns 6 to reflect selection of the "No Plots" button.

The current value of **scdb_struct->data->plot_type** is queried when the "OK" button is selected and plotting of the product is to commence.

Setting Arrow Length, Text Height and Text Expansion: The sliding bar widgest used to set arrow length, text height and text expansion are created by the function **createScaleWidgets()** via **creatProductDialog()**. **createScaleWidgets()** registers the callback functions **readArrowLen()**, **readTextHt()** and **readTextExp()** with the respective sliding bar widgets to acquire values for storage in their respective scdb_struct elements (**scdb_struct-> data->arrow_len**, **scdb-struct->data->text_ht** and **scdb_struct->data->text_exp**). The stored values will be queried when the "OK" button is selected and plotting of the product commences.

Setting the Report Type: The "Reports" portion of the "label" pop-up dialog window contains two buttons for determining the type of report ("Average File" and "Report Summary"). In addition, there are two buttons for selecting report contents ("No. of Obs" and "No. + %"). The "Average File" button is linked to the callback function **getAverageFlag()** and the adjacent textbox is linked to the callback function **getAverageName()**. The "Report Summary" button is linked to the callback function **getReportFlag()** and the adjacent textbox is linked to the callback function **getSummaryName()**. **getAverageName()** and **getSummaryName()** retrieve file names entered from the keyboard into their respective textbox widgets, storing them in **scdb_struct->data->avgFile** and **scdb_struct->data->summFile**, respectively. These values will be retrieved by functions that do the processing when the "OK" button is selected and report generation commences. The "No of Obs" button is linked to the callback function **getObsFlag()**. The "Number + %" button is linked to the callback function **getPercentFlag()**. Both the "No of Obs" button and the "Number + %" buttons are radio buttons, only one of which can be active at any time. **getObsFlag()** and **getPercentFlag()** set **scdb_struct->data->listOpt** to zero (0) and one (1), respectively. The listOpt flag will be queried when the "OK" button is selected to commence report generation.

Creating Plots and Reports: The "OK" button launches procedures that generate plots and reports based on the configuration established by user interaction with the application. Plot and report generation is the main purpose of the SCDBMS application. When the "OK" button is selected, the callback function **doneProduct()** is executed. **doneProduct()** first changes the cursor to a watch via the function **SCDB_cursor2watch()** and calls the FORTRAN function **scdb_scdb2tmp_()** with latitude/longitude and month ranges and file name information. If the data source has been established as a file, its name is passed to **scdb_scdb2tmp_()**; otherwise,

writeData2File() is employed to export SCDB data to a file. **scdb_scdb2tmp_()** creates a file containing SCDB observations in the format used to calculate averages. This file is then passed to the FORTRAN subroutine **scdb_avge_()**, which produces a file containing averages of surface current data within the given grid area. **scdb_avge_()** calls FORTRAN subroutines **scdb_stdev_()** and **scdb_setdir_()**. **scdb_stdev_()** calculates the standard deviation of a data set. **scdb_setdir_()** sets the direction of those surface currents which correspond to North, East, South and West. The file output of **scdb_avge_()** is in the format of the Average Report and can be displayed or printed directly if this option has been selected. if a Summary Report has been requested, doneProduct() next executes the FORTRAN subroutine **scdb_stat_()**, which produces a summary file based on either (1) the number of observations, or (2) number of observations including percentages. At this point, if the "No Plot" button is active, **doneProduct()** releases control to **XtMainLoop()**. If a plot has been selected, it is either created by **createDrawingArea()** (if "Screen" output is specified) or **drawHardcopy()** (if "Printer" or "Print to File" is specified). All plots are produced by the function **drawPlot()**.

Viewing and Printing the Legend: When the "Show Legend" button is selected, the function **showLegend()** is executed. In turn, **showLegend()** executes **exposeLegend()** which calls **drawLegend()** to perform the actual legend plot creation. When the "Print Legend" button is selected the function **printLegend()** is executed. **printLegend()** executes **drawHardcopy()**, which executes **drawLegend()** with output assigned to the printer (vice the screen or a file).

Closing the Dialog: Closing the log is accomplished by selecting the "Exit" button. The dialog is closed by the callback function **exitProduct()**.

4.1.3.1.9 PRODUCTS: Print Log (SCDBA Only)

The "Print Log" menu item responds only to the SCDBA. The "Print Log" dialog window, Figure 10, is produced by the callback function **selectPrintLog()** which calls the function **printLogLayout()** to construct the necessary window components and register the necessary callback functions as follows:

readFunction() - to capture the SCDB transaction function type(s) selected from the scrollable "Functions" list.
readUserName() - to capture the name of the user of interest to the SCDBA.
readMinDate() - to capture the minimum date as entered in the "Min Date" field.
readMaxDate() - to capture the maximum date as entered in the "Max Date" field.
to_fileTogglePress() - to process SCDBA selection of the "File" output option.
executeLogData() - to process SCDBA selection of the "OK" button.
resetLogStruct() - to process SCDBA selection of the "Reset" button.
quitPrintLog() - to process SCDBA selection of the "Exit" button.

fileTogglePress() calls the function **filenameDialog()** to create a secondary pop-up dialog window for retrieval of a user-entered filename by the callback function **readFilename()**. **executeLogData()** calls the DMM function **scdbLogData()** to retrieve log entries that meet the criteria established by SCDBA entries in the "Print Log" dialog window and output the data to screen, printer or file (see Figure 11).

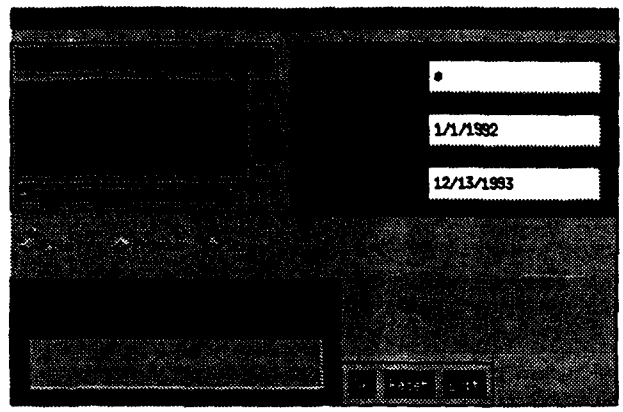


Figure 10. Print Log Definition Window.

```
-------------------------------------------------------------------------
|        Log Type          User Name    Func. Id      Date          |
|-----------------------------------------------------------------------|
|                                                                       |
|          Updated Source     ecmopdba     5       8/17/1994         |
|          Updated Source     ecmopdba     5       8/18/1994         |
|     Added Classification    ecmopdba     6       7/27/1994         |
|   Updated Classification    ecmopdba     6       7/28/1994         |
|     Added Classification    ecmopdba     6       8/9/1994          |
|            Added Source     ecmopdba     5       8/10/1994         |
~
"/u/a/scdba/scdb/src/409185088" 10 lines, 800 characters
```

Figure 11. Print Log Screen Output Example.

4.1.3.1.10 UTIL: Read Defaults File

The function **createUtil()** registers the callback function **selectReadFile()** to create the "Read Defaults File" pop-up dialog window shown in Figure 12. This dialog window is a standard Motif widget created by the function **XmCreateFileSelectionDialog().** File/directory selection from the scrollable listings, entry of a path/file into the "Selection" textbox and selection of the "Directory" button are handled internally by the widget. The callback function **fileNameAcceptCB()** responds when the "Load" button is selected by calling the function **executeReadDefaultsFile(). executeReadDefaultsFile()** resets SCDBMS default values to those contained in the newly loaded defaults file via the functions **readScdbDefaults(), clearscdbStruct(), clearScdbSelection()** and **setScdbSelection().**

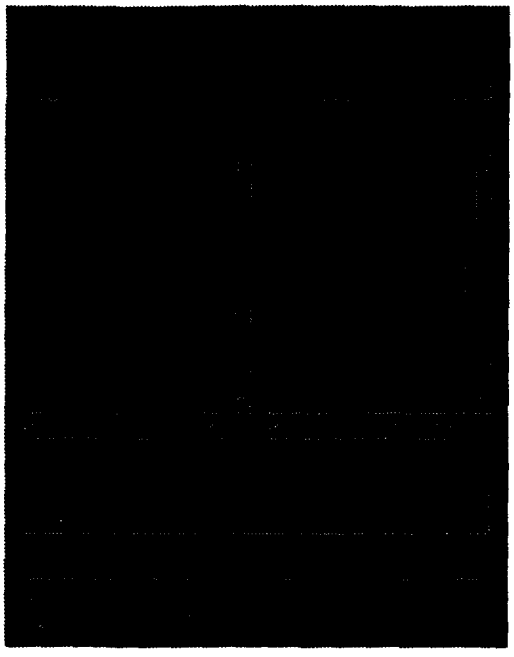


Figure 12. Util Read Defaults Window.

4.1.3.1.11 UTIL: Write Defaults File

The function **createUtil()** registers the callback function **selectWriteFile()** to create the "Write Defaults File" pop-up dialog window shown in Figure 13. **selectWriteFile()**, in turn calls the function **createFileNameDialog()** which registers the additional callback functions **exitDefaultsFile()** and **executeWriteDefaultsFile()** to handle the "Exit" and "OK" buttons, and the callback function **readFileName()** to capture the output filename entered by the user. **executeWriteDefaultsFile()** collects current values for all default variable members and writes them to a text file.

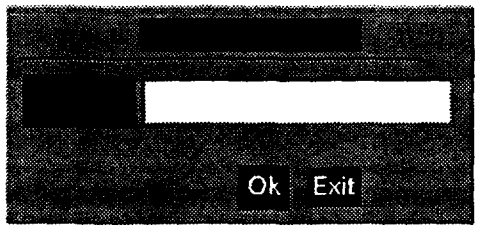


Figure 13. Util Read Defaults Window.

4.1.3.1.12 HELP

The function **createMenubarOptions()** registers the callback function **scdbOpnHelp()** to create pop-up dialog windows in response to selection of the "Help" menu header. **scdbOpnHelp()**, in turn, registers the callback function **moodsOpnHelpOk()** to process selection of the "OK" button associated with each "Help" pop-up dialog window. **scdbOpnHelp()** registers a recursive callback to itself to process a series of help pop-up dialog windows. Help text is contained in the include file **scdbOpnHelp.h**.

4.1.3.2 The Data Selection Screen

The "Data Selection" portion of the "Main Window" occupies most of the left side of the SCDBMS Main Window display screen (see Figure 2). The function **dataSelAndDisplay()** is called by the function **CreateTree()** to create both the "Data Selection" and "Selection Status" windows. **dataSelAndDisplay()** relies upon the function **dataSelectDesign()** to construct the "Data Selection" window and its suite of interactive widgets. **dataSelectDesign()** registers the following callback functions to process user selection and entry of key SCDBMS database query parameters, all of which must be defined prior to any database query:

readMonth() - to capture the minimum or maximum month selected from the scrollable "Months" listbox.

readSource()* - to capture the minimum or maximum data source code selected from the scrollable "Source" listbox.

readClass()* - to capture the minimum or maximum classification code selected from the scrollable "Classification" listbox.

The statement...

if (scdb_struct->query_val->selection_type == MINIMUM)

determines if the value retrieved by the callback functions represents a minimum or maximum value for each parameter. **scdb_struct->query_val->selection_type** is set by the callback functions **minTogglePress()** and **maxTogglePress()** which are registered by **selectDisplayDesign()**, the function which is ultimately responsible for creating the "Selection Status" window located to the right of the "Data Selection" window. Maximum and minimum values for the key parameters are placed in the structure **scdb_struct** where they may be accessed by other SCDBMS functions.

4.1.3.3 The Selection Status Screen

The "Selection Status" portion of the "Main Window" occupies most of the right side of the SCDBMS Main Window display screen (see Figure 2). The function **dataSelAndDisplay()** is called by the function **CreateTree()** to create both the "Selection Status" and "Data Selection" windows. **dataSelAndDisplay()** relies upon the function **selectDisplayDesign()** to construct the "Selection Status" window and its suite of interactive widgets. **selectDisplayDesign()** registers the following callback functions to retrieve and display minimum and maximum values for the key parameters defined within the "Data Selection" window:

minTogglePress() - to set data selection mode to minimum in response to clicking the "Minimum" toggle button.

maxTogglePress() - to set data selection mode to maximum in response to clicking the "Maximum" toggle button.

readMinTime() - to retrieve the minimum time from the **scdb_struct** structure.

readMaxTime() - to retrieve the maximum time from the **scdb_struct** structure.

readMinLon() - to retrieve the minimum longitude from the **scdb_struct** structure.

readMaxLon() - to retrieve the maximum longitude from the **scdb_struct** structure.

readMinLat() - to retrieve the minimum latitude from the **scdb_struct** structure.

readMaxLat() - to retrieve the maximum latitude from the **scdb_struct** structure.

The non-callback function **getMonthName()** is invoked to retrieve the minimum or maximum month value from the **scdb_struct** structure. The remaining maximum and minimum parameter values are directly retrieved from the **scdb_struct** structure and inserted into their text widgets by the function **selectDisplayDesign()**.

4.1.3.4 Remarks

The "Remarks" label button and its accompanying textbox are created and displayed by the function **CreateTree()**. Messages are printed in the "Remarks" textbox by the function **printmesg()** and removed by the function **clearmesg()**.

4.2 CSC-2 - The SCDBMS Data Management Module (DMM)

Visible functionality of the SCDBMS is the responsibility of the GUI. The DMM responds to internal requests from the GUI, the DAM or other DMM functions. A request may require data retrieval from the SCDB. It may require creation of a new process or application of an algorithm on the data after it is retrieved from the database. The DMM may be called upon to read SCDB data from an import file and ingest it into the SCDB. Effectively, the user is insulated from the functionality of the DMM, seeing only the results which are displayed by the GUI. Any data processing chores performed by the DMM are associated with the need to 1) provide data in specified formats to the GUI and/or DAM, or 2) ingest/retrieve data to/from the supporting SCDB. The CSU-level design of the DMM is embodied in those functions which interact with the GUI, DAM and the supporting database.

4.2.1 DMM Interaction with the GUI

With two exceptions (see Section 4.2.2, below), all communications with the DMM and its underlying database is invoked by the GUI. At the CSU level, DMM interaction with the GUI is accomplished via the following DMM functions:

InitScdbStruct() - to allocate memory for the structure **scdb_struct** and initialize values. Default data values are obtained from the **scdb.defaults** file by the function **readScdbDefaults()** and passed to GUI functions such as **selectDisplayDesign()** which display the values within its "Selection Status" text widgets.

exportData2File() - called by the GUI function **importexportOkSelect()** to query retrieve data from database and write it to a file.

readIngest_file() - called by the GUI function **importexportOkSelect()** to import SCDB ADMIN data into database with latitude, longitude and time plus seven additional variables.

executePrintHeader() - called by **outputFormatDialog()** to fetch data from the database for use by the Print Header menu option.

executeLogData() - called by **printLogLayout()** to retrieve specific log entries from the database as defined by the SCDBA.

4.2.2 DMM Interaction with the DAM

As indicated in Figure 3, the DMM and DAM are not routinely required to communicate directly in order to support SCDBA-specific actions relative to database maintenance. In most cases, the communication is accomplished indirectly; i.e., the GUI serves as the messenger between the DAM and the DMM. The following DMM functions are exceptions in that they do communicate directly with the DAM:

scdbUpdateQltv_nfo() - called by the DAM function **changeTable()** to update the SCDB qualitative information table and make the appropriate entries in the SCDB Log table.

scdbLogData() - called by the DAM function **executeLogData()** to retrieve log data from the database.

4.2.3 DMM Interaction with the SCDB

DMM functions that interact with the SCDB and their purpose are as follows:
lltn_opn() - to open the SCDB for input, output or update.
scdbUpdateQltvInfo() - to update the SCDB qualitative information table and makes the appropriate entry in the SCDB Log.
mk_dset_name() - to create the dataset name and sequence type.
llt5_wr() - to write a primary SCDB data record to the database.
lltn_wr_as() - to write an associative record to the SCDB for an ingested dataset.
llt5_rd() - to read a primary SCDB data record from the database.
llt5_ctxt() - to set the context (range of time, latitude, longitude, etc.) for records to read.

4.3 CSC-3 - The SCDBMS Data Administration Module (DAM)

The purpose of the DAM is to support 1) SCDB Administrator (SCDBA) control of the SCDBMS and 2) to exclude user access to unauthorized data. User access criteria can only be established by the SCDBA, which emphasizes the critical security responsibilities of the SCDBA. The DAM interacts directly with the GUI in support of database and software maintenance requirements. Aside from the two exception noted in Section 4.2.2, there is no direct link between the DAM and the DMM. Instead, interaction between the DAM and the DMM is accomplished via the GUI.

4.3.1 DAM Interaction with the GUI

At the CSU level, the following DAM functions interact with the GUI to perform the tasks indicated:
getUlogin() - called by **main()** to obtain user identification of the current process.
getDba() - called by **main()** to identify the database creator.
checkAccess() - called by **importexportOkSelect()** and **exportData2File()** to ensure verify user access permissions for file read/write operations.
changeTable() - called by **insCodeType()**, **updateCodeType()** and **delCodeType()** to make necessary changes to SCDBA-controlled parameter values (classification and source).

4.3.2 DAM Interaction with the DMM

(see Section 4.2.2)

5 SCDBMS DATA

5.1 The SCDB

The SCDB is described in Appendix C.

5.2 SCDBMS User Configuration Data

User configuration data is contained in the **scdb.defaults** file. An example **scdb.defaults** file can be found in Appendix G.

6 SCDBMS DATA FILES

The SCDBMS retrieves data from the SCDB, data import files and the user configuration file. There are no other data requirements for the SCDBMS.

7 REQUIREMENTS TRACEABILITY

SCDBMS functional requirements (SFR) and design requirements (SDR) have been defined for the SCDBMS. SFR/SDR requirements are also indicated in paragraphs 3.2.1, 3.2.2 and 3.2.3 for cross-reference and traceability. CSC responsibility for achieving each SFR and SDR are indicated parenthetically in the following descriptions for each SFR and SDR.

SCDBMS FUNCTIONAL REQUIREMENTS (SFR)

SFR1: Operate in an interactive manner; i.e., displays must be interactive. (GUI)
SFR2: Interactively identify geographical region boundaries. (GUI)
SFR3: Interactively identify time and date ranges. (GUI)
SFR4: Interactively identify specific classification codes and ranges of same. (GUI)
SFR5: Interactively identify specific cruise identification numbers and ranges of same. (GUI)
SFR6: Interactively identify specific data source codes and ranges of same. (GUI)
SFR7: Interactively select specific month constraints and ranges of same. (GUI)
SFR8: Interactively enter specific data loading dates and ranges of same. (GUI)

SFR9: Interactively toggle between selection of minimum and maximum retrieval parameter criteria. (GUI)
SFR10: Interactively select combinations of temperature, salinity and/or sound speed for processing. (GUI)
SFR11: Provide on-line help to users. (GUI)
SFR12: Provide on-line addition, deletion and update capability to the SCDBA for classification codes, cruise identification numbers and data source codes. (GUI/DAM)
SFR13:On-line data import capability for SCDBA. (GUI/DMM/DAM)
SFR14: On-line data export capability for users. (GUI/DMM)
SFR15: Error feedback via message facility. (GUI/DMM/DAM)
SFR16: Capability for multiple configuration/initialization files. (CSCI)
SFR17: Interactively indicate completion of retrieval criteria development and commencement of data retrieval operation based on the indicated criteria. (GUI/DMM)
SFR18: Overlay data distribution plots as points on a geographic map window. (GUI/DMM)
SFR19: Produce histogram display of data density. (GUI/DMM)
SFR20: Display depth versus parameter (temperature, salinity, sound speed) plots. (GUI/DMM)
SFR21: Display temperature versus salinity plots. (GUI/DMM)
SFR22: Output header information to printer, screen or file. (GUI/DMM)
SFR23: Maintain log of user activity with ability to print same to printer, screen or file.(GUI/DMM)
SFR24: Output statistical summary of data retrieved from database with ability to print same to printer, screen or file. (GUI/DMM)
SFR25: Provide SCDB data editing capability. (independent of SCDBMS)
SFR26: System must be able to graphically plot temperature and salinity profiles onto a geographic grid or X-Y plot. (GUI)
SFR27: Access to SCDB ASCII format. (DMM)
SFR28: Access to SCDB ADMIN format. (DMM)
SFR29: Access to Coastlines/Shorelines. (DMM)

SFR30: Selectively evaluate and/or edit SCDB data. (GUI/DMM)
SFR31: Selectively retain subsets of environmental data after evaluation and/or editing procedures have been performed. (DMM)
SFR32: Interactive termination of application. (GUI)
SFR33: Interactive reset of data retrieval parameters. (GUI)

SCDBMS DESIGN REQUIREMENTS (SDR)

SDR1: The SCDBMS must operate as a stand-alone system. (CSCI)
SDR2: The SCDBMS must operate within the UNIX operating system environment. (CSCI)
SDR3: The SCDBMS must execute within the X-Windows client-server model. (CSCI)
SDR4: Window displays must incorporate the Open Software Foundation (OSF) Motif Widget Library. (CSCI)
SDR5: An independent relational database management system (rdbms) specifically for SCDB utilization. (DMM)
SDR6: The Naval Environmental Operational Nowcast System (NEONS) will be incorporated as the interface to the rdbms. (DMM)
SDR7: The SCDBMS must contain internal links to the SCDB for data import/export/ retrieval. (DMM)
SDR8: The SCDBMS must incorporate detection of user privilege levels based on data classification codes. (GUI)

8 NOTES

A glossary of terms is contained in Appendix A. A listing of acronyms is contained in Appendix B. Appendix C is the SCDBMS RDBMS Specification. Appendix D lists SCDB functional (DBFR) and design (DBDR) requirements. Appendix E discusses the SCDBMS development environment. Appendix F is a listing of SCDB structures. Appendix G provides an explanation of the user default file.

APPENDIX A. GLOSSARY OF TERMS

Computer Software Configuration Item (CSCI) - a software application or a major component thereof.

Computer Software Component (CSC) - a top level functional module within a computer software configuration item (CSCI). CSC's are generally considered to be one structural level below the CSCI.

Computer Software Unit (CSU) - low level software modules, usually at the function or subroutine level that perform specific functions within a CSC.

Data Administration Module (DAM) - the SCDBMS module responsible for SCDBA security and administrative maintenance functions.

Data Management Module (DMM) - the SCDBMS module responsible for database access.

Graphical User Interface (GUI) - the SCDBMS module responsible for interfacing with the user and controlling the graphical and display functionality of the SCDBMS.

Widget - "... a graphic device capable of receiving input from the keyboard and the mouse and communicating with an application or another widget by means of a callback. Every widget is a member of only one class and always has a window associated with it." (from: OSF/Motif Programmer's Guide, Rev. 1.1, Open Software Foundation, Prentice Hall, Englewood Cliffs, NJ, 1991, p. GL-13.)

APPENDIX B. LIST OF ACRONYMS

ANSI - American National Standards Institute
CAST - Center for Air Sea Technology
CSC - Computer Software Component
CSCI - Computer Software Configuration Item
CSU - Computer Software Configuration Unit
DAM - Data Administration Module
DBDR - Database Design Requirement
DBFR - Database Functional Requirement
DMM - Data Management Module
DOD - Department of Defense
GUI - Graphical User Interface
IDEAS - Interactive Data Editing and Analysis System
MDBA - MOODS Database Administrator (alias SCDBA)
MOODS - Master Oceanographic Observation Data System
MSU - Mississippi State University
NASA - National Aeronautics and Space Administration
NAVOCEANO - Naval Oceanographic Office
NEONS - Navy Environmental Operational Nowcast System
OSF - Open Software Foundation
RDBMS - Relational Database Management System
SCDB - Surface Currents Database
SCDBA - Surface Currents Database Administrator
SCDBMS -Surface Currents Database Management System
SDR - SCDBMS Design Requirement
SFR - SCDBMS Functional Requirement
SQL - Structured Query Language

APPENDIX C. THE SCDB RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) SPECIFICATION

The SCDBMS accesses the SCDB RDBMS, a modified NEONS database created specifically to support the CSCI. The SCDB exists as a dedicated external and independent element of the system, accessible via embedded SQL queries to the underlying RDBMS engine. The SCDBMS queries the database and retrieves requested data from it by calling NEONS software library functions. NEONS provides a data model for all generic data types (grid, volume, latitude/longitude/time (LLT), line, grid, image and geographical); however, the SCDBMS only uses a modified version of the latitude/longitude/time data type to store SCDB profiles, plus the geographical data type which stores coastlines and ocean bathymetry datasets. SCDB profile datasets can be retrieved by the following parameters:

> latitude, longitude, date/time, month, water depth, parameter, source code, instrument, classification code, and cruise identification number.

These parameters represent an enhancement of NEONS capability to meet SCDBMS requirements. The NEONS latitude/longitude/time data type allows queries only by latitude, longitude and date/time. The additional parameters (month, water depth, parameter, source code, instrument, classification code, and cruise identification number) were moved from the RDBMS binary large object storage format to allow faster dataset specification and retrieval. The SCDBMS-specific version of the LLT data type is referred to as "LLTN_7" because of the seven additional query parameters. The SCDBMS version of NEONS also incorporates special relational tables to support security classification codes and a SCDB transaction log.

Further information about NEONS, its structure and use is available in the NEONS design document ["Database Design Document for the Naval Environmental Operational Nowcast System", Version 3.5, Naval Research Laboratory (NRL), Monterey, CA 93943-5006]. In addition, the NEONS installation package includes source code and manual pages for each major functional element. NEONS version control is managed by the Naval Research Laboratory, Monterey, CA.

APPENDIX D. FUNCTIONAL AND DESIGN REQUIREMENTS FOR THE SCDB RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

DATABASE FUNCTIONAL REQUIREMENTS (DBFR):

DBFR1: Database must be capable of data retrieval.
DBFR2: Data ingestion into the rdbms is to be accomplished either internally or externally to the SCDBMS.
DBFR3: The searchable database attributes must include month, water depth, parameter, source code, instrument type, classification and cruise identification number.

DATABASE DESIGN REQUIREMENTS (DBDR):

DBDR1: RDBMS engine is Empress.
DBDR2: RDBMS model is the Naval Environmental Operational Nowcast System (NEONS) Version 3.5.1.
DBDR3: Database must be compatible with NAVOCEANO Program Modernization Initiative (PMI).

APPENDIX E. THE SCDBMS DEVELOPMENT ENVIRONMENT

The SCDBMS has been developed to operate within a Sun Microsystems SparcStation model 10 computer hardware environment. The operating system is SUNOS version 4.1.3, including the resident SUN C compiler which was used to write the SCDBMS software code. Some minor elements of NEONS have been written in FORTRAN 77 (Sun FORTRAN 77 version 1.4). Graphics support is provided by UNIRAS ag/X Toolmaster version 6v4a. The RDBMS engine is Empress version 6.2. The windowing environment consists of X-Windows version X11R5 and the OSF Motif widget set version 1.3.

APPENDIX F. SCDB STRUCTURES

A C structure is a collection of data variables, pointers or other structures grouped together for the convenience of using a single variable name to reference or identify the whole group. The use and behavior of structures is covered in any credible C programming language tutorial or textbook.

1. **The NAVO Structure** - comprises the critical data framework of the SCDBMS application. The **NAVO** structure and its internal structures are defined as follows:

```
typedef struct {
        STATUS_BOARD_WIDGETS *status_board_widgets;
        QUERY_VAL     *query_val;
        TIME_LOC        *LocTime;
        Widget          region, globe_map;
        char            seq_type[31];
        char            vrsn_name[31];
        char            defaultsFile[120];
        } NAVO;
```

Structures Internal to the NAVO Structure

```
typedef struct {
        Widget lat_min_text, lat_max_text;
        Widget lon_min_text, lon_max_text;
        Widget time_min_text, time_max_text;
        Widget class_min_text, class_max_text;
        Widget month_min_text, month_max_text;
        Widget source_min_text, source_max_text;
        Widget min_toggle, max_toggle;
        } STATUS_BOARD_WIDGETS;

typedef struct {
        double lat_min_val, lat_max_val;
        double lon_min_val, lon_max_val;
        DATE  time_min_val, time_max_val;
        double hour_min_val, hour_max_val;
        long  class_min_val, class_max_val;
        long  month_min_val, month_max_val;
        long  source_min_val, source_max_val;
        Boolean selection_type;
        } QUERY_VAL;

typedef struct {
        DATE        start_date;
        DATE        end_date;
        double      min_lat;
        double      min_lon;
        double      max_lat;
        double      max_lon;
        int         start_hour;
        int         end_hour;
```

```
int         start_min;
int         end_min;
int         start_sec;
int         end_sec;
} TIME_LOC;
```

Note: The Widget structure is proprietary to the XLib/MOTIF libraries.

2. **The DATE Structure** - defines the elements which comprise date and time within the SCDBMS.

```
typedef struct {                /* date structure */
    int year;                       /* year number since 0 BC */
    int month;                      /* month number in year */
    int day;                        /* day number in month */
    int type;                       /* time coordinate type */
} DATE;
```

3. **The DISTRIBUTION Structure** - groups all variables required to plot a histogram or accumulation chart.

```
typedef struct {
    float           lat_deg;        /* latitude  interval */
    float           lon_deg;        /* longitude  interval */
    int             lon_binsize;    /* longitude cell count */
    int             lat_binsize;    /* latitude cell count */
    Boolean         grid;           /* To indicate if grid is needed */
    char            coast_type[15]; /* 3km, 8km, 20km */
    char        title1[50];     /* Plot type title */
    char        title2[50];     /* User-supplied title */
    int             dev_type;       /* Screen/window */
    int             plot_type;      /* Plot/Histogram/Accumulation */
    int             rec_num;        /* Num of Obs Read */
    Boolean         thereisaacclpix;/* Indicates presence of Pixmap */
    Boolean         thereisahistpix;/* Indicates presence of Pixmap */
    Boolean         thereisapix;    /* Indicates presence of Pixmap */
    Pixmap              pixmap;             /* Pixmap to store graphics */
    XWindowAttributes pix_at;     /* Store dimensions of Pixmap */
    XWindowAttributes hist_pix_at; /* Store dimensions of Pixmap */
    XWindowAttributes accl_pix_at; /* Store dimensions of Pixmap */
    Pixmap              hist_pixmap;    /* Pixmap to Histogram */
    Pixmap              accl_pixmap;    /* Pixmap to Accumulation plot */
    float           pix_width;      /* Size of Display Pixel */
    float           *hist_array;    /* To keep count in cells */
    Boolean         dataset_selected;/* Avoid reselecting data for Hist*/
    Widget              coast_text;     /* Indicate choice Made */
    NAVO                *moods_struct;  /* Global Structure */
    Window              accl_window;    /* Window ID */
    Window      hist_window;    /* Window ID */
    Window      plot_window;    /* window ID */
} DISTRIBUTION;
```

4. **The LOGSTRUCT Structure** - groups the necessary variables to output a portion of the SCDBMS transaction log.

```
typedef struct {
        int   log_type;
        Widget list;
        Widget to_screen;
        Widget to_file;
        Widget to_printer;
        Widget name_text;
        Widget min_date_text;
        Widget max_date_text;
        Widget msg_text;
        char  userName[20];
        DATE  min_date;
        DATE  max_date;
        long  func_id;
        long  min_func_id;
        long  max_func_id;
        char  filename[80];
        } LOGSTRUCT;
```

5. **The SMRYPAGE Structure** - groups the necessary variables to output a summary page.

```
typedef struct {
        float min_lat;
        float min_lon;
        float max_lat;
        float max_lon;
        struct {
                int yr;
                int ttl_yr_cnt;
                struct {
                        int mo;
                        long ttl_mo_cnt;
                        } mo_bin[12];
                } yr_bin[MAX_CNT];
        struct {
                long src_code;
                long ttl_src_cnt;
                } src_bin[MAX_CNT];
        struct {
                long inst_code;
                long ttl_inst_cnt;
                } inst_bin[MAX_CNT];
        struct {
                int clas_code;
                long ttl_clas_cnt;
                long ttl_qual_cnt;
                long ttl_dist_cnt;
```

```
        long ttl_dist_qual_cnt;
        } clas_bin[MAX_CLAS_CNT];
} SMRY_PAGE;
```

6. **The IMPORTEXPORT Structure** - contains the necessary variablers to import or export a data file.

```
typedef struct {
        NAVO        *moods_struct; /* Global Structure */
        char  filename[80];   /* Filename to store header */
        int     type;         /* IMPORT,EXPORT, or EXPORT_IDEAS  */
        Widget ok;            /* Ok Widget */
        } IMPORTEXPORT;
```

APPENDIX G. THE SCDBMS DEFAULT CONFIGURATION FILE

Default values for parameters appearing within the "Selection Status" textboxes of the SCDBMS main display may be maintained in a user-created default file. The user may specify the file containing default values by including the "-f" flag, followed by the filename, when launching the SCDBMS (example: scdb -f myfile.def). If the default file is not indicated on the command line, the SCDBMS will look for the file "scdb.def" in the current directory. If the "scdb.def" file cannot be located, parameter boxes within the "Selection Status" portion of the main display will be blank at start up. The format for a default file is as follows:

```
Min Latitude       = 10.0
Max Latitude       = 25.0
Min Longitude      = -100.0
Max Longitude      = -70.0
Min Date           = 01/01/1900
Max Date           = 08/10/1993
Min Hour           = 0.0
Max Hour           = 23.0
Min Classification = 1100010
Max Classification = 1100010
Min Month          = 1
Max Month          = 12
Min Source         = 0
Max Source         = 15
```

The values contained in a default file are ordered in a left-to-right, top-to-bottom sequence according to their appearance within the "Selection Status" portion of the main display. The format is free form except that labels and numerical values must be separated by at least one space and each labeled line must end in a carriage return. For each labeled category, there is a single numerical entries, either a minimum and maximum value, on each line.

The numerical precision and default date formats contained in default file are not necessarily the same as formats displayed in the scrollable listboxes within the "Selection Status" or "Data Selection" portions of the SCDBMS main display screen. Note that the "Hour" entries in the "SCDB.def" file is in floating point format, which differs from the representation of the hour within the "Time" textbox in the SCDBMS main display. Also, the "Date" entries use different formats in the "scdb.def" file and the "Time" textbox within the "Selection Status" board of the SCDBMS main display.

Distribution List

1. Commanding Officer, Code N3211
   Naval Oceanographic Office
   Stennis Space Center, MS 39529
   (10 copies)

2. Technical Director
   Code OOT
   COMNAVMETOCCOM
   Stennis Space Center, MS 39529

3. Oceanographer of the Navy
   U.S. Naval Observatory
   34th and Massachusetts Avenue
   Washington, DC 20392

4. Space and Naval Warfare
   Systems Command
   Code PMW175-3B
   2451 Crystal Drive
   Arlington, VA 22245-5200

5. Defense Technical Information Center
   Building 5, Cameron Station
   Alexandria, VA 22304-6145
   (2 copies)

6. Director, Sponsored Programs Administration
   Mississippi State University
   P.O. Box 6156
   Mississippi State, MS 39762

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. Agency Use Only *(Leave blank).*	2. Report Date.	3. Report Type and Dates Covered.
		TECHNICAL NOTE

4. Title and Subtitle.

DESIGN DOCUMENT FOR THE SURFACE CURRENTS DATA BASES (SCDB) SYSTEM (SCDBMS) VERSION 1.0

5. Funding Numbers.

Program Element No.

Project No.

Task No.

Accession No.

6. Author(s).

RAMESH KRISHNAMAGARU
M.S. FOSTER
CHERYL CESARIO
VISHNUMOHAN DAS

7. Performing Organization Name(s) and Address(es).

MISSISSIPPI STATE UNIVERSITY
CENTER FOR AIR SEA TECHNOLOGY
BUILDING 1103, ROOM 233
STENNIS SPACE CENTER, MS 39529

8. Performing Organization Report Number.

CAST TECHNICAL NOTE 8-94

9. Sponsoring/Monitoring Agency Name(s) and Address(es).

NAVAL OCEANOGRAPHIC OFFICE (CODE N3211)
STENNIS SPACE CENTER, MS 39529

10. Sponsoring/Monitoring Agency Report Number.

CAST TECHNICAL NOTE 8-94

11. Supplementary Notes.

Research performed via Mississippi Research Consortium under NASA procurement Office Contract NAS13-330, Delivery Order #53.

12a. Distribution/Availability Statement.

Approved for public release; distribution is unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

The Surface Currents Database Management System (SCDBMS) provides access to the Surface Currents Database (SCDB) maintained by NAVOCEANO. This manual provides the design document and database specification for the SCDBMS.

14. Subject Terms.

(U) Design Document (U) SCDBMS (U) SCDB
(U) CAST (U) GUI (U) NAVOCEANO (U) NEONS

15. Number of Pages.

41

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	